VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Integrace SIP Proxy Kamailio a Media/RTP Proxy a zjištění výkonnosti tohoto řešení

SIP Proxy and Media/RTP Proxy Integration and its Performance Assesment

2014 Jakub Budoš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Jakub Budoš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Integrace SIP Proxy Kamailio a Media/RTP Proxy a zjištění výkonnosti tohoto řešení
SIP Proxy and Media/RTP Proxy Integration and its Performance Assesment

Zásady pro vypracování:

Kamailio jako jedna z nejúspěšnějších open-source implementací SIP Proxy představuje zajímavou alternativu k pobočkové ústředně Asterisk a to zejména v prostředích vyžadujících vysoký výkon, vysokou míru zabezpečení a flexibilitu, kde zároveň nejsou pokročilé služby pobočkové ústředny třeba. Cílem bakalářské práce je instalace, základní konfigurace a otestování výkonnosti SIP Proxy Kamailio ve spolupráci s Media/RTP Proxy a zhodnocení zjištěných poznatků.

Body zadání:
1. SIP Proxy Kamailio, architektura a funkce.
2. Media/RTP Proxy a její využití.
3. Instalace a konfigurace SIP Proxy Kamailio s Media/RTP Proxy.
4. Otestování výkonnosti daného řešení.
5. Zhodnocení výsledků.

Seznam doporučené odborné literatury:

[1] Goncalves, F. E.: Building Telephony Systems with OpenSER: A step-by-step guide to building a high performance Telephony System. Packt Publishing, 2008, ISBN: 978-1847193735.
[2] Internetová dokumentace projektu Kamailio, dostupná na adrese: http://www.kamailio.org/w/

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Rozhon**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárné pramene a publikácie, z ktorých som čerpal.

V Ostrave 6.5.2014 .....

Na tomto mieste by som rád poďakoval Ing. Janovi Rozhonovi za odbornú pomoc a za konzultácie pri zostavovaní tejto bakalárskej práce.

Abstrakt

Cieľom tejto bakalárskej práce je integrovať open source SIP proxy server Kamailio a modul Mediaproxy alebo RTPproxy a zistiť vplyv tohto modulu na výkonnosť serveru. Teoretická časť obsahuje základné informácie o VoIP, signalizačných protokoloch H.323, SIP a prenosovom protokole RTP. Ďalej obsahuje kapitoly, ktoré popisujú Kamailio, jeho históriu, architektúru, informácie o RTPProxy a Mediaproxy moduloch. Bakalárska práca obsahuje detailný postup inštalácie, kompletnú konfiguráciu Kamailia a modulu RTPproxy. Na záver sa práca zaoberá záťažovým testovaním serveru a zhodnotením dosiahnutých výsledkov.

Klúčové slova: VoIP, SIP, Kamailio, RTPproxy, RTP, konfiguračný súbor, záťažové testovanie, kodek

Abstract

The goal of this bachelor thesis is to integrate open source SIP proxy server Kamailio and RTPproxy or Mediaproxy module and measure the impact of this module on the performance of the server. The theoretical part contains basic information about VoIP, signaling protocols H.323, SIP and RTP transport protocol. This part also contains chapters that describe Kamailio, its history, architecture, information about RTPproxy module and Mediaproxy module. Bachelor thesis contains a detailed installation procedure, complete configuration of Kamailio and RTPproxy module. The last chapter deals with server stress testing and evaluation of the results achieved.

Keywords: VoIP, SIP, Kamailio, RTPPRoxy, RTP, configuration file, stress testing, codec

Zoznam použitých skratiek a symbolov

API	–	Application Programming Interface
CPU	–	Central Processing Unit
DNS	–	Domanin Name System
GIT	–	GNU Interactive Tools
GPG	–	GNU Privacy Guard
GPL	–	General Public License
H.245	–	signalizačný protokol
H.323	–	protokoly pre audio-vizuálne relácie
HTTP	–	Hypertext transfer Protocol
ICMP	–	Internet Control Message Protocol
IETF	–	Internet Engineering Task Force
IP	–	Internet Protocol
ISDN	–	Integrated Services Digital Network
ITU-T	–	International Telegraph Union-Telecommunication
kbit/s	–	kilobit per second
kHz	–	kiloHertz
LAN	–	Local Area Network
ms	–	millisecond
NAT	–	Network Address Translation
OSI	–	Open Systems Interconnection
Q.931	–	signalizačný protokol
QOS	–	Quality Of Service
RAS	–	Remote Access Service
RFC	–	Request for Comments
RTCP	–	Real-time Transport Control Protocol
RTP	–	Real-time Transport Protocol
SCTP	–	Stream Control Transmission Protocol
SDP	–	Session Description Protocol
SIP	–	Session Instantion Protocol
SMTP	–	Simple Mail Transfer Protocol
SQL	–	Structured Query Language
SRTP	–	Secure Real-time Transport Protocol
TCP	–	Transport Control Protocol

TLS	–	Transport Layer Socket
UAC	–	User Agent Client
UAS	–	User Agent Server
UDP	–	User Datagram Protocol
VOIP	–	Voice Over Internet Protocol
WAN	–	Wide Area Network

Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

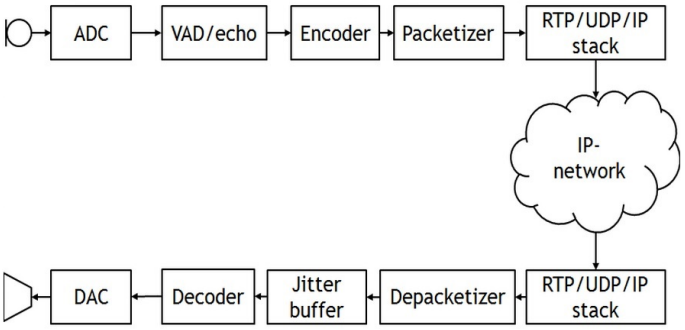
1 Úvod

Táto bakalárska práca sa zoberá VoIP technológiou. Prvá časť práce je teoretická a obsahuje kapitoly venujúce sa VoIP technológií a jej protokolom. Jedná sa hlavne o signalizačné protokoly H.323 a SIP a prenosový protokol RTP. Práca sa podrobne zaoberá SIP serverom Kamailio. Existuje množstvo softvérových pobočkových ústrední, ktoré ponúkajú veľa užitočných funkcií. Medzi najpopulárnejšie open source ústredne patrí Asterisk. Kamailio je vhodnou alternatívou najmä v situáciách, kedy nie sú pokročilé funkcie Asterisku potrebné, ale je potreba zaistiť vysoký výkon, spoľahlivosť a bezpečnosť. Kapitola 3 obsahuje informácie o Kamailie, podrobnejšie opisuje jeho históriu a architektúru. V tejto kapitole tiež opisujem štruktúru konfiguračného súboru. Klienti pripojení za NAT zariadeniami tvoria značnú časť užívateľov VoIP služieb. Kamailio nedokáže týmto klientom sprostredkovať komunikáciu. Tento problém rieši pomocou pridania modulov. V tejto práci sú opísane moduly RTPproxy a Mediaproxy, ktoré patria medzi tzv. far-end NAT riešenia. To znamená, že umožňujú komunikáciu klientom, ktorý sú pripojení za NAT bez toho aby títo klienti museli vykonávať dodatočnú konfiguráciu na svojich NAT zariadeniach. Tiež dokážu plniť úlohu média proxy serveru. Naviac modul RTPproxy dokáže pracovať ako brána medzi IPv4 a IPv6 sieťou a disponuje funkciami pre nahrávanie hovorov. Bližšie však popisujem modul RTPproxy, ktorý som v mojej práci použil. Druhá časť práce je praktická a obsahuje podrobný postup inštalácie a všetky potrebné konfigurácie Kamailia a RTPproxy. Ďalej opisujem úpravu smerovacej logiky Kamailia s použitím RTPproxy ako média proxy serveru. Záver práce je venovaný záťažovým testom serveru s použitím kodekov G.711, G.729, G.722 a riešeniu problémov spojených s týmito testami. Koniec práce obsahuje výsledky testov, ktoré sú zamerané na využitie CPU pri použití RTPproxy pre jednotlivé kodeky.

2 VoIP

Voice over IP je technológia, ktorá umožňuje prenášať hlas v sieťach založených na IP protokole. Oproti klasickej telefónií je výhodná najmä ekonomický. Používa dátové siete, ktoré sa v dnešnej dobe veľmi rýchlo rozširujú. Veľmi obľúbenou sa stala najmä pre geografický rozsiahle spoločnosti, ktoré ju nasadzovali na svojich vybudovaných dátových sieťach. Spolu s pripojením operátora poskytujúcim prenos hlasu na IP konektivite tak VoIP tvorí dobrú alternatívu ku klasickým telefónnym sieťam. VoIP má samozrejme aj svoje nevýhody. Tie plynú najmä z toho, že dátové siete neboli navrhnuté pre prenos hlasu v reálnom čase. Jedná sa hlavne o oneskorenie paketov alebo ich stratovosť, ozvenu, zníženú spoľahlivosť a dostupnosť služby, bezpečnostné riziká. [1, 2]

Voip používa IP protokol na tretej vrstve modelu OSI. Na štvrtej vrstve je to prevažne transportný protokol UDP. Ako komunikačný protokol k prenosu hlasu sa používa RTP protokol alebo jeho šifrovaná verzia SRTP a riadiaci protokol RTCP, ktorý nesie informácie o prenose. Ďalšou dôležitou skupinou protokolov sú signalizačné protokoly. Tieto protokoly slúžia na vytváranie, riadenie a ukončovanie spojenia vo VoIP. Do tejto kategórie patria hlavne protokoly H.323 a SIP. S príchodom SIP protokolu sa používanie H.323 utiahlo do úzadia. [2, 3]



Obr. 2.1: Typický komunikačný kanál vo VoIP[4]

Na obrázku 2.1 je zobrazený prenos hlasu vo VoIP od jedného koncového zariadenia k druhému. Pre dosiahnutie nižšieho dátové toku sa používajú kodeky. Znížením dátového toku zároveň znížime aj nároky kladené na prenosové médium. Medzi najpoužívanejšie kodeky pre prenos hlasu vo VoIP patria G.711, G.729, G.723.1. IP telefónia požaduje zabezpečenie kvality služby QoS(Quality of Service). QoS je skupina mechanizmov, ktoré sa snažia minimalizovať stratovosť paketov, oneskorené doručenie a poskytnúť konštantnú kapacitu. V dnešných sieťach sa predovšetkým používajú mechanizmy: Best-effort services, Differentiated services, Integrated services[1, 5]

2.1 Signalizačné protokoly

2.1.1 H.323

H.323 bol vyvinutý organizáciou ITU-T(International Telecommunications Union - Telecommunications). Prvá verzia tohto protokolu sa objavila v roku 1996. Spočiatku špecifikoval spôsoby prevádzky multimediálnej komunikácie v prostredí LAN sieti, ktoré neposkytujú žiadnu garanciu kvality služieb QoS. Druhá verzia bola vydaná v roku 1998 a ta už poskytovala podporu aj pre rozsiahlejšie siete WAN a Internetu. H.323 protokol bol prvým protokol, ktorý prispôsobil transportný protokol RTP pre prenos hlasu v IP sieti. [6]

Skladá sa z komponentov terminál, gateway(brána), gatekeeper, multipoint control unit(MCU).
Terminál sú koncový klienti, ktorým je poskytnutá duplexná komunikácia v reálnom čase. Je to základná a povinná jednotka H.323 siete. Všetky terminály musia podporovať RAS(Registration Admission Status), Q.931, H.245 a RTP. RAS vykonáva interakciu s gatekeeprom, Q.931 je použitý na signalizáciu a nastavenie hovoru, H.245 slúži na výmenu schopnosti terminálov a vytvorenie prenosového kanálu pre hlas a video a RTP prenáša hlasové pakety.
Gateway (brána) je koncový bod v sieti. Okrem komunikácie medzi terminálmi a bránami zriaďuje aj spojenie medzi ďalšími terminálmi ITU v prepínaných sieťach. To znamená, že dokáže do VOIP siete pripojiť aj klasické telefónne zariadenia ako fax, analógový a ISDN telefón. Ďalšie funkcie brány sú spracovanie signalizačných správ, kompresia a dekompresia hlasového signálu v reálnom čase, vytváranie a rozkladanie paketov pri prechode medzi inými druhmi sieti.
Multipoint Control Unit(MCU) – umožňuje uskutočniť komunikáciu medzi tromi a viac účastníkmi. Zároveň určuje prenosové atribúty konferencie napríklad použitie kodeku. Skladá sa z dvoch komponentov:

- Multipoint Controller(MC) – zostavuje konferenciu, zisťuje vlastnosti terminálu počas konferencie, nastavuje a ukončuje kanály pre hlasové, obrazové a dátové prenosy. Používa H.245 signalizáciu. MC nevykonáva multiplexovanie hlasu, obrazu a dát.

- Multipoint Processor(MP) – vykonáva multiplexovanie dátového toku, ktorý riadi MC. Prispôsobuje komunikáciu pre jednotlivé terminály.

Gatekeeper(správca) jeho hlavnou úlohou je riadenie celej prevádzky v sieti. Je voliteľným komponentom, avšak v jeho prítomnosti musia všetky komunikujúce zariadenia používať jeho služby. Poskytuje riadiace služby pre terminály a brány. Neprenáša užitočné dáta. Hlavné funkcie sú preklad alias adries(telefónne číslo, emailová adresa a podobné) na IP adresy, účtovanie služieb, smerovanie hovorov a ďalšie. [6, 5].

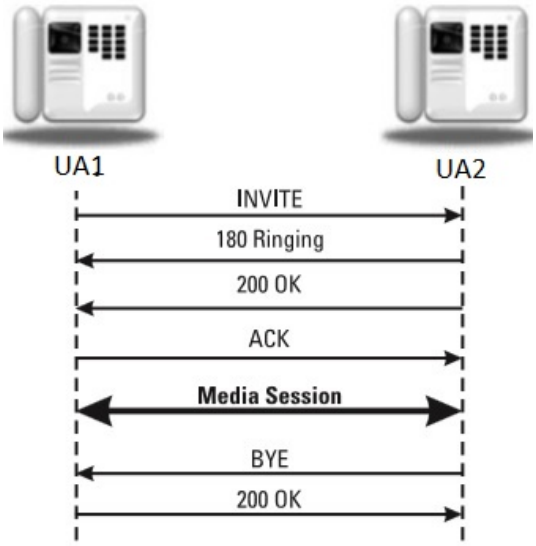
2.1.2 SIP

SIP(Session Initiation Protocol) je signalizačný protokol relačnej vrstvy modelu RM OSI. Bol vyvinutý organizáciou IETF(Internet Engeneering Task Force) a je popísaný v niekoľkých dokumentoch známych ako RFC(Request For Comments). Najbežnejšie používaná verzia je SIP verzia 2 definovaná dokumentom RFC 3261. Používa sa na vytváranie, úpravu a ukončovanie multimediálnych spojení s jedným alebo viacerými účastníkmi. Je textovo orientovaný a svojou filozofiou je veľmi podobný protokolu HTTP alebo SMTP. Rozdielom je, že SIP dokáže používať aj protokol UDP na transportnej vrstve. [7]

Komponenty SIP siete:

- User Agent je koncové zariadenie, ktoré sa stará o nadviazanie spojenia s ostatnými agentmi. Zvyčajne sa jedná o IP telefón alebo bránu.
  -User Agent Client(UAC) inicializuje SIP požiadavku o nadviazanie spojenia.
  -User Agent Server(UAS) reaguje na poslanú požiadavku a posiela odpovede.
- Redirect server používa databázu alebo lokalizačné služby aby zistil adresu používateľa. Informácie o jeho adrese pošle späť volajúcemu s odpoveďou presmerovania hovoru.
- Lokalizačný server má uložené informácie o možnom umiestnení klienta. Umiestnením sa chápe jeho adresa, číslo alebo skupina adries.
- Proxy server preposiela SIP správy od UAC. Funkciou podobný serveru presmerovania. Avšak po získaní adresy z lokalizačného servera sám nadviaže spojenie so serverom(UAS).
- Registračný server sa používa na zaregistrovanie pripojeného používateľa. Po pripojení užívateľa registračný server pridelí jeho logickej adrese adresu fyzickú. Túto informáciu potom poskytne lokalizačnej službe v sieti.[7, 5]

SIP nešpecifikuje, ktorý prenosový protokol sa použije a ani typ použitého kodeku. K účelom špecifikácie spojenia používa protokol SDP(Session Description protocol). Ten okrem toho rozhoduje aj o tom, čo sa bude prenášať(hlas, video atď.) a prenosovej rýchlosti. Zápis SDP správy v SIP je vo formáte atribút = hodnota. Medzi tieto atribúty môže patriť napríklad: verzia protokolu, názov relácie, adresa a port pôvodcu, doba počas ktorej je relácia aktívna, adresa a port média spojenia atď. Atribúty môžu byť podľa potreby k SDP protokolu pridávané. [9]



Obr. 2.2: Nadviazanie SIP spojenia[7]

Klienti v sieti SIP medzi sebou komunikujú pomocou správ, viď obrázok 2.2. Tieto správy môžu byť typu žiadosť alebo odpoveď. Odpoveď nám indikuje, či správa so žiadosťou uspela alebo zlyhala, prípadne prečo zlyhala.

Základné SIP Request (požiadavky) správy sú:

- INVITE – žiadosť o nadviazanie spojenia
- BYE – žiadosť o ukončenie spojenia
- ACK – potvrdenie spojenia
- REGISTER – žiadosť o registráciu klienta na register server
- CANCEL – žiadosť o zrušenie prebiehajúcej žiadosti INVITE
- OPTIONS – žiadosť o zaslanie podporovaných funkcií na serveri [8]

SIP Response (odpoveď) správy sú:

- 1xx – informačná odpoveď, napr. 180 ringing (vyzváňanie)
- 2xx – úspešná odpoveď, napr. 200 ok
- 3xx – odpoveď presmerovania, napr. 302 moved temporarily (dočasné presmerovanie)
- 4xx – odpoveď, že požiadavka zlyhala, napr. 404 not found (nenájdené)
- 5xx – odpoveď, že server nepodporuje službu, napr. 503 service unavailable (služba neprípustná)
- 6xx – odpoveď o globálnej chybe, napr. 600 busy everywhere (neprípustné všade) [8]

2.2 Komunikačné protokoly

2.2.1 RTP

RTP(real-time transport protocol) je určený na prenos hlasu alebo videa v reálnom čase. Používa UDP protokol na transportnej vrstve a IP protokol na sieťovej vrstve. Tento protokol nezaručuje doručenie dát a ani ich doručenie v správnom poradí. Definuje časové značky a poradové čísla paketov. Na ich základe je multimediálna aplikácia schopná rozoznať chýbajúce pakety. Formát RTP datagramu je definovaný dokumentom RFC 3550 a preto vyhovuje všetkým aplikáciám v reálnom čase. V hlavičke sa nachádzajú informácie o poradovom čísle paketu a časová značka. Ďalej informácie o formáte multimediálneho obsahu, o začiatku a konci rámca, identifikáciu zdroja a synchronizáciu pre detekciu rôzneho kolísania oneskorenia v rámci daného toku a pre potrebnú kompenzáciu pri prehrávaní hlasu alebo videa. Existuje aj šifrovaný variant SRTP a variant CRTP, ktorý používa komprimačné metódy. [10]

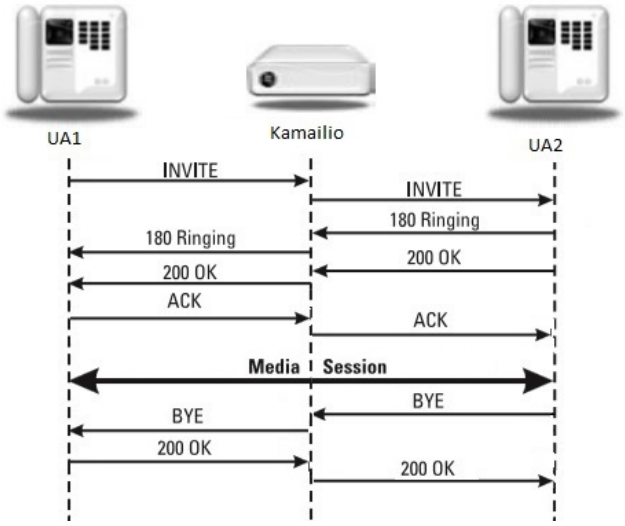
2.2.2 RTCP

RTCP(Real-time Transfer Control Protocol) je riadiaci protokol, ktorý spolupracuje s protokolom RTP. Sám o sebe nenesie žiadne dáta iba riadiace informácie o toku dát RTP protokolu. Jeho hlavnou úlohou je poskytovať spätnú väzbu na kvalitu QoS služieb. Monitoruje doručovanie dát, čo pomáha príjemcovi detegovať stratu paketov a ten potom informuje odosielateľa, ktorý vykoná kompenzáciu kolísania oneskorenia v sieti. RTCP datagramy obsahujú informácie, podľa ktorých môže vysielajúca strana dynamický meniť tok dát, napr.: rýchlosť prenosu. Poskytuje tak služby pre riadenie a kontrolu zahltenej siete. [10]

3 Kamailio

Kamailio je open source SIP server vydaný pod licenciou GPL. Slovo kamailio pochádza z havajštiny a znamená hovoriť. História Kamailia siaha do roku 2001, kedy sa začal vývoj SIP express router(SER) ako projekt FhG FOKUS Institute, Berlín, Nemecko. Samotný projekt Kamailia vznikol až v roku 2005, vtedy ešte pod názvom OpenSER, keď sa oddelil od projektu SER. OpenSER bol zameraný na vytvorenie voľného vývojárskeho prostredia, aby vytvoril silný a rozšíriteľný open-source SIP server. Z dôvodu porušovania ochranných známok sa v roku 2008 premenoval na Kamailio. Ešte v tom istom roku sa Kamailio a SER opäť spojili a začali pracovať na integrácií týchto dvoch SIP server aplikáciách. S príchodom verzie 3.0.0 o rok neskôr bola integrácia dokončená. [12]

Je napísaný v jazyku C a určený pre UNIX/LINUX systémy. Jeho architektúra je navrhnutá tak, aby poskytovala čo najvyšší výkon. Flexibilitu zabezpečuje jeho malá hardvérová náročnosť, preto je vhodný pre nasadenie od embedded zariadení až po viac jadrové serveri. Kamailio má iba základné funkcie potrebné pre fungovanie SIP serveru. Medzi jeho základné funkcie patria: registračný server, lokalizačný server, SIP proxy server, aplikačný SIP server a server presmerovania. K účelu rozšírenia funkcionality používa moduly. Modul môžeme pridať bez toho, aby sme zasahovali do jadra Kamailia a tým neznižovali jeho spoľahlivosť. Na obrázku 3.1 vidíme, že ak pridáme Kamailio do komunikácie bude sa správať ako SIP proxy server. [13]



Obr. 3.1: Nadviazanie SIP spojenia s použitím Kamailia[7]

Poznámka 3.1 Kamailio štandardne ostáva v signalizačnej trase aj počas rozpadu spojenia. Zisti tak dĺžku hovoru pre prípadne účtovanie hovoru.

3.1 Architektúra

Architektúra Kamailia bola navrhnutá tak, aby mu poskytovala čo najvyšší výkon. Kamailio používa pre komunikáciu s administrátorom kontrolné rozhranie. Existujú dva typy kontrolných rozhraní, ktoré Kamailio používa:

- MI - Managment interface - toto rozhranie nie je doporučené, preto boli jeho zdrojové kódy presunuté do interných knižníc.
- RPC- Remote procedure interface - štandardizovaný spôsob spúšťania príkazov. V súčasnosti je doporučeným kontrolným rozhraním. [12]

3.1.1 Komponenty jadra

Jadro poskytuje Kamailiu základné funkcie.

- SIP parser - Kamailio implementuje svoj vlastný SIP parser. Jedná sa o inkrementačný typ parseru. To znamená, že parsuje pokiaľ nenarazí na zhodu elementu alebo koniec SIP správy. Tento komponent tiež obsahuje triedy pre prácu so SIP správami.
- Config parser & interpreter - Používa utility flex a bison k parsovaniu konfiguračného súboru a vytvára strom akcií, ktorý je spustený za behu pri prijatí každej SIP správy.
- SIP transport layer - Implementuje funkcie pre podporu transportných protokolov UDP, TCP, TLS, SCTP. Implementuje aj podporu pre DNS.
- Memory manager - Stará sa rozdelenie dostupnej pamäte. Pracuje s privátnou a zdieľanou pamäťou. Privátna pamäť je špecifická pre jednotlivé procesy. Používa sa pre premenné, ktoré nemusia byť viditeľné v iných procesoch. Dáta uložené v zdieľanej pamäti sú viditeľné pre všetky moduly. Zdieľaná pamäť sa inicializuje až po analýze konfiguračného súboru.
- Locking manager - V prípade zdieľania zdrojov sa používajú uzamykacie mechanizmy. Koreňovým elementom je mutex semafor. Uzamykanie môže byť riešené aj ako lokálna premena alebo pole jednoduchých zámkov. [12]

3.1.2 Interné knižnice

Komponent zavedený od verzie 3.2.0. Jedná sa o súbor funkcií napísaných v jazyku C. Sú automatický načítane za behu programu v prípade, ak niektoré moduly potrebujú použiť funkcie z nich. Boli tu presunuté časti Database API a Managnent interface API starého jadra Kamailia verzie 1.5.x. Ďalej sa tu nachádza štatistický engine, ktorý sleduje stav a vyťaženie Kamailia. Zavedenie interných knižníc má svoje výhody. Presunutím funkcií sa zmenšila veľkosť jadra. Tím sa stal vhodnejší pre embedded zariadenia a ten istý kód z viacerých modulov môže byť raz uložený v internej knižnici, tím predídeme duplikovaným častiam kódu. [12]

3.1.3 Moduly

Modul je samostatný program, ktorý rozširuje funkcionalitu Kamailia. Jeho pripojenie nevyžaduje zásah do jadra a tím sa nenarušuje jeho stabilita. Existuje viac ako 150 modulov a s každou novou verziou Kamailia pribúda podpora nových modulov. Môžeme pridať moduly s funkciami ako sú napríklad: účtovanie hovoru, autorizáciu a autentifikáciu, registračný a lokalizačný manažment, prepojenie užívateľov za NAT, prepojenie s SQL alebo non-SQL databázami, prepojenie s PSTN a mnoho ďalších. [12]

3.1.4 Konfiguračný súbor

Je hlavným súborom, ktorý riadi činnosť Kamailia. Skladá sa z dvoch častí. Inicializačná časť sa vykoná iba raz pri spustení. Obsahuje nastavenie globálnych parametrov napríklad IP adresu a port, kde bude Kamailio prijímať SIP správy alebo nastavenie domény, viď nižšie.

```
alias="sip.mydomain.com"
```

```
listen=udp:10.0.0.10:5060
```

Ďalej obsahuje zoznam pridaných modulov. Modul pridáme zadaním kľúčového slova *loadmodule*, za ktorý napíšeme názov modulu v úvodzovkách, viď nižšie.

```
loadmodule "mi_fifo.so"
```

Potom nasleduje časť, kde sa nastavujú parametre modulov pri spustení. Parameter nastavíme tak, že najprv zadáme kľúčové slovo *modparam* a do zátvorky uvedieme názov modulu, názov parametru a jeho hodnotu, viď nižšie.

```
modparam("mi_fifo", "fifo_name", "/tmp/kamailio_fifo")
```

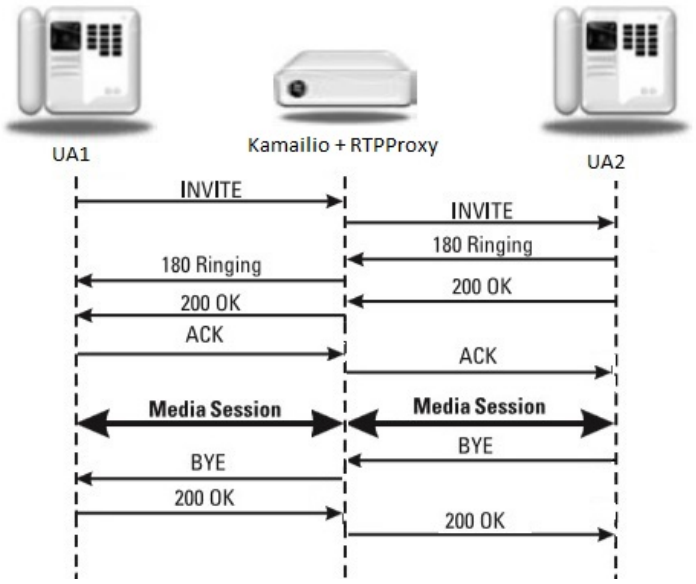
Ďalšia časť konfiguračného súboru sa nazýva runtime. Tá sa vykoná vždy po prijatí SIP správy. Ide o smerovaciu logiku Kamailia, ktorá určuje akým spôsobom sa prijatá SIP správa spracuje a aká funkcia sa vykoná. Je napísaná v jazyku C a od verzie 3.0 používa preprocesorové direktívy. To umožňuje definovať časti, ktoré môžeme jednoducho vypnúť alebo zapnúť. Smerovacia logika sa skladá z blokov:

- request route - hlavný smerovací blok, kontroluje spracovanie každej SIP správy
- onreply route - spracováva SIP odpovede
- failure route - spracováva SIP negatívne odpovede
- route - vlastný nadefinovaný blok, pracuje ako podprogram [8]

4 Moduly Mediaproxy a RTPproxy

4.1 RTPproxy modul

RTPproxy je modul určený pre prácu s RTP streamami. Je to samostatný softvér, ktorý spolupracuje so SIP proxy servermi akým je Kamailio. Tento modul vyvinul Maxim Sobolev a dnes je aktívne udržovaný spoločnosťou Sippy Software, Inc. Hlavný dôvodom vzniku tohto modulu bol problém SIP proxy serverov, ktoré nedokázali umožniť komunikáciu užívateľom za NAT zariadeniami. Hlavnou úlohou RTPproxy je teda umožniť komunikáciu užívateľom, ktorý sa nachádzajú za NAT zariadeniami bez toho aby museli vykonať dodatočnú konfiguráciu na svojich NAT zariadeniach. Tento modul môže byť nakonfigurovaný ako média proxy alebo ako brána medzi IPv4 a IPv6 sieťami. Tiež disponuje funkciami pomocou, ktorých dokáže nahrávať hovory. RTPproxy úzko spolupracuje s modulom nathelper. V jeho spolupráci môžeme v sieti použiť viac inštancií RTPproxy nainštalovaných na rozdielnych serveroch. To nám umožni efektívne rozložiť záťaž a zvýšiť spoľahlivosť. Ak pridáme RTPproxy do komunikácie všetky média budú smerovať cez server s RTPproxy, viď obrázok 4.1. [15, 14]



Obr. 4.1: SIP spojenie s použitím Kamailia a RTPproxy[12]

4.1.1 Princíp činnosti RTPproxy

Keď Kamailio príjme SIP INVITE správu zistí hodnotu parametru call-id a prepošle ju na soket, kde počúva RTPproxy. RTPproxy po prijatí správy zisťuje, či spojenie s takýmto call-id existuje. Ak existuje vráti Kamailiu UDP port pre toto spojenie, ak neexistuje, tak vytvorí nové spojenie a priradí mu prvý voľný UDP port z nadefinovaného rozsahu portov. Potom ho pošle späť Kamailiu. To následné prepíše media ip a port v SDP správe, aby RTP datagramy smerovali na RTPproxy. Keď Kamailio príjme SIP odpoveď opäť získa call-id a pošle RTPproxy. Tento krát RTPproxy iba hľadá spojenie s takýmto call-id a nevytvára nové. Ak ho nenájde vráti odpoveď v podobe chybového hlásenia. Ak spojenie existuje, pošle späť Kamailiu UDP port pre toto spojenie. To po prijatí informácie opäť prepíše media ip a port v hlavičke SDP, aby RTP datagramy smerovali na RTPproxy. Po vytvorení spojenia RTPproxy počúva na porte, ktorý bol pridelený tomuto spojeniu a čaká na prijatie aspoň jedného UDP datagramu. Po prijatí datagramu z jednej z komunikujúcich strán prepíše jeho ip adresu a port na RTPproxy ip adresu a port ako zdroj tohto datagramu. To isté vykoná pre datagram prijatý druhou stranou. Potom RTPproxy začne preposielať RTP datagramy medzi komunikujúcimi stranami. [15]

4.2 Mediaproxy modul

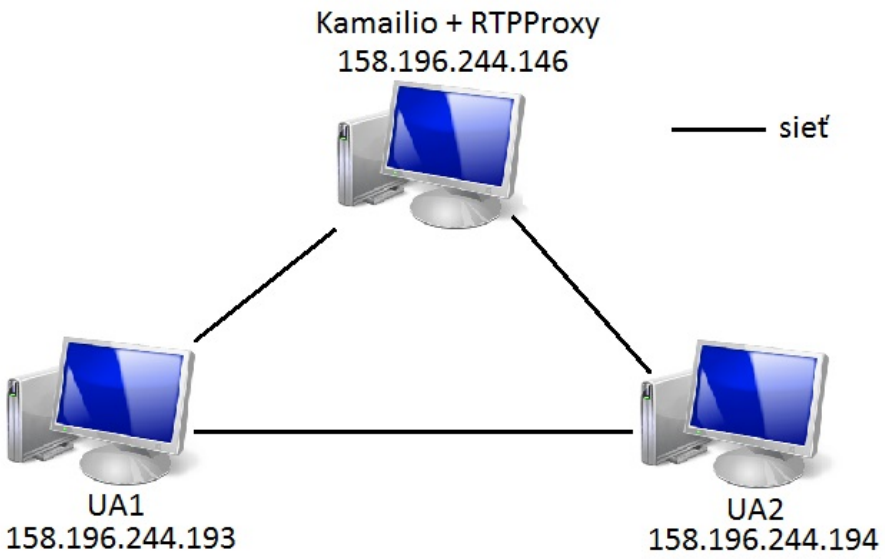
Mediaproxy je obdobou modulu RTPproxy. Je teda primárne určený na prepojenie užívateľov za NAT zariadeniami. Jeho koncept je založený na myšlienke mať niekoľko média proxy serverov v doméne pre vyrovnanie záťaže a pre prípad zlyhania jedného zo serverov sa môže použiť iný dostupný média proxy server. K tomu bola prispôsobená architektúra tohto modulu, ktorá sa skladá z dvoch častí. Dispečer beží na rovnakom zariadení ako SIP proxy a jeho úlohou je vybrať vhodný média proxy server pre komunikáciu. Média proxy server môže bežať na rovnakom zariadení ako dispečer alebo na viacerých zariadeniach v sieti. Úlohou média proxy serveru je sprostredkovanie samotnej komunikácie medzi zúčastnenými stranami. Podmienkou aby mohol média proxy server fungovať je, aby malo zariadenie, na ktorom beží nastavenú verejnú IP adresu. Média proxy server môže so SIP proxy komunikovať aj priamo bez použitia dispečera. [17]

4.3 Porovnanie RTPproxy a Mediaproxy

Oba moduly ponúkajú funkcie ako je prepojenie NAT klientov a vytvorenie média serveru. RTPproxy má však viac funkcií. Dokáže nahrávať RTP spojenia a plniť úlohu mostu medzi IPv4 a IPv6 sieťami, čo Mediaproxy nedokáže. [16, 18] V tejto práci som použil modul RTPProxy. Rozhodol som sa tak najmä z dvoch dôvodov. Prvým je podpora zo strany Kamailia. Štandardný konfiguračný súbor obsahuje zakomentované časti, ktoré popisujú ako pridať a spustiť RTPproxy. Druhým dôvodom bol jazyk, v ktorom sú tieto moduly napísane. RTPproxy je napísaný v jazyku C preto dokáže spracovávať inštrukcie niekoľko krát rýchlejšie ako Mediaproxy napísaný v jazyku python [8]

5 Integrácia Kamailia a RTPproxy

Hlavnou náplňou tejto práce je nainštalovať Kamailio spolu s RTPproxy a nakonfigurovať ich k vzájomnej spolupráci. Kamailio aj RTPproxy sú výhradne určené len pre unixové alebo linuxové operačné systémy. Prvým krokom je teda zabezpečiť zariadenie s takýmto systémom. V mojom prípade som mal k dispozícii vzdialený virtuálny server s priradenou IP adresou 158.196.244.146. Aby som mohol svoju konfiguráciu otestovať potreboval som ďalšie dva virtuálne serveri, ktoré predstavovali klientov v mojej topológií. Tie som mal dostupné na IP adrese 158.196.244.193-4. Na každom serveri bol nainštalovaný operačný systém linux, konkrétne išlo o distribúciu Ubuntu 13.04, 64bit. Táto distribúcia už má nainštalované všetky doplnkové balíčky potrebné pre chod Kamailia. Topológiu môžeme vidieť na obrázku 5.1.



Obr. 5.1: Zapojenie experimentálnej topológie

5.1 Inštalácia Kamailia

Kamailio som nainštaloval pomocou linuxového balíčka. Ďalším spôsobom môže byť inštalácia z komprimovaného súboru. Najprv som si stiahol GPG kľúč, aby som mohol balík s Kamailiom nainštalovať. Ten som potom pridal do zoznamu kľúčov v linuxe pomocou príkazov:[11]

```
wget http://deb.kamailio.org/kamailiodebkey.gpg
```

```
apt-key add kamailiodebkey.gpg
```

Kamailio som potom musel pridať do zoznamu balíčkov, ten vlastne určuje to, kde v sieti sa daný balíček nachádza. Tento zoznam bol v mojej distribúcií linuxu uložený v súbore */etc/apt/sources.list*. Súbor je v textovej podobe, ktorý otvoríme pomocou nejakého textového editoru. V ňom som pridal daný záznam o balíčku. Existujú balíčky vo viacerých verziách, podľa toho akú verziu linuxu používame. Moja verzia má označenie precise, preto som pridal záznam:[11]

```
deb http://deb.kamailio.org/kamailio precise main

deb-src http://deb.kamailio.org/kamailio precise main
```

V tomto prípade sa mi stiahne aktuálne posledná stabilná verzia 4.1, okrem toho si môžeme vybrať aj nižšie verzie alebo devel verziu. Devel verzia je nestabilná verzia, ktorá je určená pre testovanie nových funkcií Kamailia. Ďalším krokom je už samotné stiahnutie a inštalácia Kamailia. To som vykonal pomocou príkazu:[11]

```
apt-get install Kamailio
```

Kamailio nedisponuje vlastnou databázou. Databáza je potrebná k tomu, aby sme do nej mohli ukladať údaje o užívateľoch, poprípade iné údaje spojené s Kamailiom. Kamailio podporuje SQL databázy, napríklad Oracle, SQLite, MySQL. Ja som si vybral MySQL. Túto databázu som nainštaloval z balíčka príkazom:

```
apt-get install mysql-server
```

5.2 Inštalácia RTPproxy

Inštalácia RTPproxy je oveľa jednoduchšia. Opäť som si vybral inštaláciu pomocou linuxového balíčka. V prípade RTPproxy nemusíme pridávať záznam ani sťahovať kľúč. Posledná verzia je 1.2.1 a nainštaloval som ju príkazom:

```
apt-get install rtpproxy
```

5.3 Konfigurácia RTPproxy

Konfigurácia RTPproxy vyžaduje iba nastavenie komunikačného soketu. Kamailio aj RTPproxy máme nainštalované na jednom zariadení, preto som použil unix soket. Všetky parametre som nastavil pomocou prepínačov pri spustení RTPproxy:

```
rtpproxy -s unix:/var/run/rtpproxy.sock -u kamailio
```

Prepínač -s nastavuje komunikačný soket RTPproxy. Na tomto sokete bude prebiehať komunikácia s Kamailiom. Prepínač -u udáva to, kto bude používateľom tohto procesu. Druhou možnosťou je nastavenie parametrov v konfiguračnom súbore RTPproxy. Ten sa nachádza v */etc/default/rtpproxy*.

5.4 Konfigurácia Kamailia

Po nainštalovaní je Kamailio vo vypnutom stave. Tento stav je definovaný v súbore */etc/default/kamailio*. V tomto súbore musíme nájsť riadok:

```
RUN_KAMAILIO=yes
```

Tento riadok je štandardne zakomentovaný alebo je hodnota nastavená na false. V mojom prípade stačilo tento riadok odkomentovať, aby sa Kamailio mohlo spustiť. Ďalším krokom je nakonfigurovať spoluprácu Kamailia a databázy MySQL. V súbore */etc/kamailio/kamctlrc* musíme nastaviť SIP doménu a typ databázy, ktorú chceme použiť. Ak používame DNS služby môžeme namiesto IP adresy zadať názov tejto domény. V tomto súbore som nastavil svoju IP adresu a databázu, ktorú som použil:

```
SIP_DOMAIN=158.196.244.146
```

```
DBENGINE=MYSQL
```

Kamailio používa databázu pre ukladanie informácií. V našom prípade ide hlavne o uchovanie informácií o klientoch. Vytvorenie databázy som vykonal príkazom:

```
kamdbctl create
```

V tejto databáze sa štandardne vytvoria dvaja užívatelia kamailio a kamailioro. Nastavenia tejto databázy môžeme upraviť v súbore */etc/kamailio/kamctlrc* pred tým, ako ju vytvoríme.
Do takto vytvorenej databázy môžeme pridávať užívateľov. Formát príkazu je:

```
kamctl add meno heslo
```

Aby sme mohli takto užívateľa pridať potrebujeme práva k zápisu do databázy. Po spustení tohto príkazu musíme zadať heslo užívateľa s právom zápisu. V našom prípade ide o užívateľa kamailio s heslom kamailiorw.
Takto nakonfigurované Kamailio som spustil príkazom:

```
service kamailio start
```

Teraz máme nastavenú základnú konfiguráciu Kamailia. Jeho funkčnosť môžeme otestovať pridaním dvoch užívateľov do databázy a použitím dvoch SIP softvérových telefónov.

Mojím cieľom je však nakonfigurovať Kamailio tak, aby spolupracovalo s modulom RTPproxy. Najskôr som si otvoril hlavný konfiguračný súbor Kamailia, ktorý je umiestnený v */etc/kamailio/kamailio.cfg*. Potom som pridal modul RTPproxy do Kamailia. To som vykonal načítanim modulu:

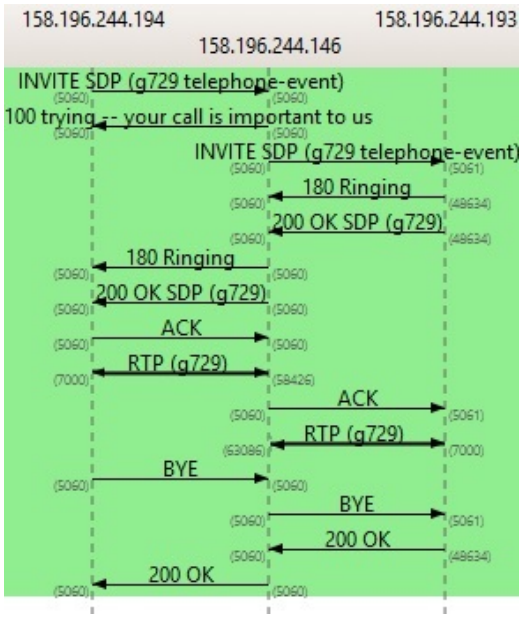
```
loadmodule "rtpproxy.so"
```

V kapitole 5.3 som nastavil komunikačný soket RTPproxy. O tomto sokete musíme informovať Kamailio, aby vedelo kam posielať správy určené pre RTPproxy. To som spravil nastavením parametru pred spustením Kamailia. V konfiguračnom súbore som pridal riadok:

```
modparam("rtpproxy", "rtpproxy_sock",
        "unix:/var/run/rtpproxy/rtpproxy.sock")
```

5.5 Modifikácia smerovacej logiky

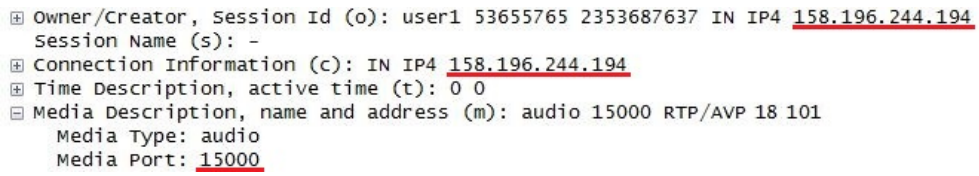
Smerovacia logika definuje to, ako sa každá prijatá SIP správa spracuje a aká funkcia sa vykoná. V mojom prípade musím túto logiku upraviť tak, aby sa modul RTPproxy správal ako média proxy server. To znamená, že všetky RTP datagramy budú preposielané cez server s RTPproxy. Tento modul má naimplementované funkcie, ktoré používa k tomu, aby upravoval smerovaciu logiku Kamailia. Tieto funkcie sú detailne zdokumentované na webových stránkach Kamailia [16].



Obr. 5.2: Zachytená komunikácia v programe Wireshark

Na obrázku 5.2 je zachytená komunikácia v mojej experimentálnej topológií pomocou programu Wireshark. Prvou správou, ktorú klient posiela aby nadviazal spojenie je SIP INVITE správa. Táto správa vo svojom tele nesie správu SDP protokolu. V tejto SDP správe sú informácie, na základe ktorých sa vytvorí mediálne spojenie. Nás predovšetkým

zaujímajú informácie obsiahnuté v parametroch *connection information(c)* a *media port*. Parameter *Owner/Creator(o)* obsahuje IP adresu zakladateľa relácie, viď obrázok 5.3.



Obr. 5.3: Fragment SDP správy odoslanej z UAC

Preto musíme použiť jednu z dostupných funkcií modulu RTPproxy, ktoré sa postarajú o prepísanie týchto parametrov. V hlavnom smerovacom bloku request_route sa najprv otestuje zmysluplnosť SIP správy v sekundárnom bloku route(REQINIT) a potom zavolám funkciu rtpproxy_manage(), viď nižšie.

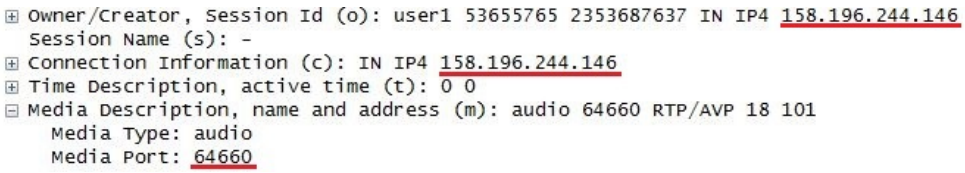
```
request_route {
        route(REQINIT);
        rtpproxy_manage("co");
        if (is_method("CANCEL"))
        {
                if (t_check_trans()) {
                        route(RELAY);
                }
                exit;
        }
        route(WITHINDLG);
        t_check_trans();
        remove_hf("Route");
        if (is_method("INVITE|SUBSCRIBE"))
                record_route();
        if (is_method("INVITE"))
        {
                setflag(FLT_ACC);
            }
        route(SIPOUT);
        route(REGISTRAR);
        if ($rU==$null)
        {
                sl_send_reply("484","Address Incomplete");
                exit;
        }
        route(LOCATION);
}
```

Výpis 5.1: Volanie rtpproxy_manage() v request_route bloku

Táto funkcia v sebe kombinuje funkcie modulu RTPproxy:

- rptproxy_offer() prepisuje parametre v tele SDP správy pri prijatí správy INVITE.
- rtpproxy_answer() prepisuje parametre v tele SDP správy pri prijatí správy 200 OK.
- unforce_rtp_proxy() ukončí spojenie RTPproxy [16].

Tieto funkcie používa podľa toho, aká správa sa aktuálne spracováva. Ak je to správa INVITE použije funkciu rptproxy_offer(), v prípade správy 200 OK rtpproxy_answer() a v prípade správ BYE a CANCEL unforce_rtp_proxy(). Túto funkciu som použil s flagmi 'co' vo formáte rtpproxy_manage("co"). Flagy zapínajú niektoré funkcie modulu alebo bližšie špecifikujú čo má táto funkcia vykonať. Flag 'c' prepisuje hodnotu parametru *connection information(c)* a flag 'o' hodnotu parametru *Owner/Creator*[16]. Táto funkcia sa postará o to, že sa IP adresa parametru *connection information(c)* a *Owner/Creator(o)* prepíše z pôvodnej IP adresy klienta na IP adresu serveru s RTPproxy. RTPproxy vytvorí pre toto spojenie port, na ktorom sa uskutoční prenos médií. Tento port potom nahradí pôvodný port, ktorý určil volajúci klient, viď obrázok 5.4.



Obr. 5.4: Fragment SDP správy odoslanej z Kamailia

Takto upravená SDP správa s novou adresou a portom sa odošle z Kamailia v tele SIP INVITE správy volanému klientovi. Ten ju spracuje a pošle odpoveď. Ak spojenie chce nadviazať pošle späť Kamailiu SIP správu 200 OK. Táto správa opäť obsahuje vo svojom tele správu SDP protokolu s informáciami o mediálnom spojení. V nej musíme opäť prepísať IP adresu parameteru *connection information(c)*, *Owner/Creator(o)* a port parametru *media port*. Znovu som použil funkciu rtpproxy_manage("co") s tým rozdielom, že som ju zavolal v bloku onreply_route, ktorý sa stará o spracovanie pozitívnych SIP odpovedi, viď nižšie.

```
onreply_route[MANAGE_REPLY] {
        if (status=~"[12][0-9][0-9]")
                rtpproxy_manage("co");
}
```

Výpis 5.2: Volanie rtpproxy_manage() v onreply_route bloku

Takto upravená správa sa pošle volajúcemu klientovi. Potom sa medzi klientmi a RTPproxy vytvorí spojenie a začne sa prenos RTP datagramov. Ak hovor skončí RTPproxy automatický ukončí svoju činnosť. Avšak v prípade ak Kamailiu dorazí negatívna SIP správa musíme sa postarať o ukončenie RTPproxy. Tieto správy sa spracovávajú v bloku failure_route, v ktorom som zavolal funkciu rtpproxy_manage("co"), viď nižšie.

```
failure_route [MANAGE_FAILURE] {
        rtpproxy_manage("co");
        if (t_is_canceled())
                exit;
}
```

Výpis 5.3: Volanie rtpproxy_manage() v failure_route bloku

6 Testovanie a Analýza

6.1 Testovanie výkonnosti serveru

Testovanie som vykonal v mojej experimentálnej topológií na virtuálnom serveri s nainštalovaným Kamailiom a RTPproxy. Na tomto virtuálnom serveri som sledoval aký vplyv má použitie modulu RTPproxy na využitie CPU. Testy som previedol tri krát vždy pre iný kodek. Na odporúčanie vedúceho práce som zvolil kodeky G.711 A-law, G.729 a podľa vlastného výberu G.722. Technické detaily týchto kodekov sú v tabuľke 6.1. Aby som vôbec mohol testovať potreboval som na serveri vytvoriť zaťaženie v podobe hovorov. Na to som použil testovací nástroj SIPp.

Kodek	Algoritmus	Vzorkovacia frekvencia (kHz)	Bitová rýchlosť (kbit/s)	Latencia(ms)
G.711	A-law	8	64	0,125
G.729	CS-ACELP	8	8	15
G.722	ADPCM	16	64	4

Tabuľka 6.1: Technické detaily použitých kodekov

SIPp je nástroj pre testovanie výkonu. Pracuje so SIP protokolom a je založený na princípe klient-server. Klient predstavuje User Agent Client(UAC), ktorý inicializuje SIP spojenie. Server predstavuje User Agent Server(UAS), teda volaného účastníka. Jeho funkciou je teda simulovať SIP spojenia. SIPp generuje správy podľa scenárov. Scenár je xml súbor, v ktorom sú nadefinované formáty SIP správ, ktoré sa budú odosielať. SIPp dokáže posielať aj média(audio, video), čo je veľmi užitočné ak chceme testovať modul RTPproxy [19].

Na zvyšných dvoch virtuálnych serveroch som stiahol SIPp v3.4 a nainštaloval. SIPp má niekoľko implicitných scenárov. Tieto scenáre však nefungovali s použitím Kamailia. To bolo spôsobené tým, že Kamailio používa funkciu record_route(), ktorú implicitné scenáre nepodporuju. Táto funkcia udržiava Kamailio v signalizácií až do skončenia hovoru, teda preposiela správy ukončujúce hovor. To je užitočné v prípadoch ak chceme zistiť dĺžku hovoru napríklad pre účtovanie alebo pre štatistické účely. Nefunkčnosť týchto scenárov som vyriešil tak, že som si vytvoril vlastne scenáre, tak aby pracovali s funkciou record_route(). V scenári pre UAC som najprv nastavil parameter *rrs* na hodnotu true po odoslaní INVITE správy, kedy sa očakávajú odpovede 180, 183, 200, viď nižšie.

```
<recv response="180" rrs="true" optional="true"></recv>
<recv response="183" rrs="true" optional="true"></recv>
<recv response="200" rrs="true" rtd="true"></recv>
```

Parameter rrs znamená Record route set: uloží hlavičku prijatej SIP správy, ktorú potom môžeme zavolať pomocou kľúčového slová [routes]. Parameter *rrs* je nastavený v tagu *recv*. Tento tag indikuje to, aká prijatá SIP správa sa očakáva. Parameter response potom

udáva na akú konkrétnu odpoveď sa čaká[19]. Túto uloženú hlavičku som potom zavolal v správach ACK a BYE, viď nižšie.

```
ACK [next_url] SIP/2.0 [routes]
```

```
BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0 [routes]
```

Potom som musel upraviť aj UAS scenár. V ňom som opäť nastavil parameter *rss* na hodnotu true po prijatí SIP správy INVITE, viď nižšie.

```
<recv request="INVITE" rrs="true" crlf="true"></recv>
```

Ďalej som pridal k správe 200 OK, ktorá obsahuje aj SDP správu kľúčové slovo [last_Record-Route:], v ktorom je uložená hlavička SIP správy, ktorá sa uložila po prijatí SIP INVITE správy. To isté som pridal aj pre správu 200 OK bez SDP správy[19].

Druhou možnosťou bolo odstrániť funkciu record_route z Kamailia, čo je však v reálnom nasadení nepraktické. Na začiatok som si vytvoril zdroj RTP datagramov v podobe pcap súborov. Ten som vytvoril odchytením 45 sekundového hovoru v programe Wireshark. Takto som vytvoril pcap súbor pre každý kodek. Použitie týchto kodekov je špecifikované v SDP správe. Do nej som potom pridal v UAC aj UAS scenári v SDP správe tieto kodeky, viď nižšie.

```
m=audio [auto_media_port] RTP/AVP 8 9 18
a=rtpmap:18 G729/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:9 G722/16000
```

Po odoslaní ACK správy, ktorá potvrdzuje vytvorenie spojenia som nastavil posielanie médií. Na odporučenie vedúceho práce som zvolil dĺžku hovoru 45 sekúnd, ktorý určuje parameter pause, viď nižšie.

```
    <nop>
      <action>
        <exec play_pcap_audio="pcap/g729.pcap"/>
      </action>
    </nop>
  <pause milliseconds="45000"/>
```

Výpis 6.1: Posielanie médií v SIPp

Takto upravené scenáre by stačilo potom už len spustiť. Tu som však narazil na problémy. Jedným problémom bolo obmedzenie počtu otvorených súborov v linuxe. Tento limit mi nedovoľoval dosiahnuť maximálneho vyťaženia serveru, pretože bol nastavený na hodnotu 1 024. Pre každý hovor sa vytvára soket, ktorý predstavuje jeden otvorený súbor. Ak k tomu pripočítame sokety, ktoré sa vytvárajú pre signalizáciu dosiahneme hranicu približne 500 súbežných hovorov. Toto obmedzenie som musel manuálne nastaviť na vyššiu hodnotu na všetkých troch virtuálnych serveroch príkazom:

```
ulimit -n 100000
```

Druhým problémom boli iptables. SIPp UAC implicitne pre odosielanie a prijímanie RTP datagramov používa port 6 000. V mojom prípade však na tomto porte nepočúval. To spôsobilo, že prichádzajúcu komunikáciu nemal kto spracovať a kernel musel odpovedať prostredníctvom ICMP správ. Množstvo týchto správ začalo značne vyťažovať procesor až ho úplne zahltilo. Riešením bolo zakázať prichádzajúcu komunikáciu na porty v rozsahu 7 000 - 65535, viď nižšie.

```
iptables -A INPUT -p udp --dports 7000:65535 -j DROP
```

Najprv som spustil SIPp server, pretože ak by som spustil skôr SIPp klienta, hovory ktoré vygeneruje pred spustením serveru nebudú spracované. Server som spustil príkazom:

```
sipp -sf UAS.xml -i 158.196.244.193:5061 -rtp_echo
```

Prepínač -sf určuje použitie vlastného scenáru, prepínač -i nastaví média IP adresu a port v SDP správe. V prípade ak ho nepoužijeme SIPp použije lokálnu adresu 127.0.0.1. Prepínač -rtp_echo spôsobí to, že prijaté RTP datagramy pošle späť odosielateľovi. Takto sa postaráme o to, že komunikácia prebehne oboma smermi. Potom som už len spustil SIPp klienta príkazom:

```
sipp -sf UAC.xml 158.196.244.193:5061 -i 158.196.244.194:5061
-rsa 158.196.244.146:5060 -skip_rlimit -mp 7000
```

Prvá adresa hovorí o tom, kde sa nachádza SIPp server, teda volaný klient. Prepínač -rsa určuje IP adresu a port vzdialeného serveru, ktorý preposiela SIP správy. V mojom prípade je to server s Kamailiom. Prepínač -skip_rlimit odstraňuje limit maximálne vytvorených soketov. Limit je 1 024, preto by som bez tohto parametru nebol schopný dosiahnuť vyšší počet súbežných hovorov. Prepínač -mp nastaví číslo lokálneho soketu pre mediálne spojenie. Tu použijem číslo portu, pre ktorý som zakázal prichádzajúcu komunikáciu.

6.2 Analýza výsledkov

Virtuálny server, ktorý som testoval mal 2,1 GHz procesor, k dispozícií 2 GB operačnej pamäte a nainštalovaný operačný systém Ubuntu 13.04, 64 bit. Stav CPU som sledoval v programe sysstat. Pre každý kodek som si zvolil 10 meracích bodov. Každý bod predstavoval určitý počet súbežných hovorov. Napríklad ak som dosiahol 860 hovorov, tak som túto hodnotu rovnomerné rozdelil na 10 meraní, teda každých 86 súbežných hovorov som vykonal meranie. Konkrétny počet súbežných hovorov som dosiahol použitím prepínaču -m, ktorý nastaví maximálný počet hovorov. Implicitne SIPp vytvára každú sekundu 10 hovorov. Takto by som nikdy nedosiahol viac ako 450 súbežných hovorov, pretože po 45 sekundách hovor skončí. Prepínačom -r som zvýšil počet vytvorených hovorov na 100 za 1 sekundu. Potom príkaz pre SIPp klienta bude vyzerať následovne:

```
sipp -sf UAC.xml 158.196.244.193:5061 -i 158.196.244.194:5061
-rsa 158.196.244.146:5060 -skip_rlimit -mp 7000 -m 860 -r 100
```

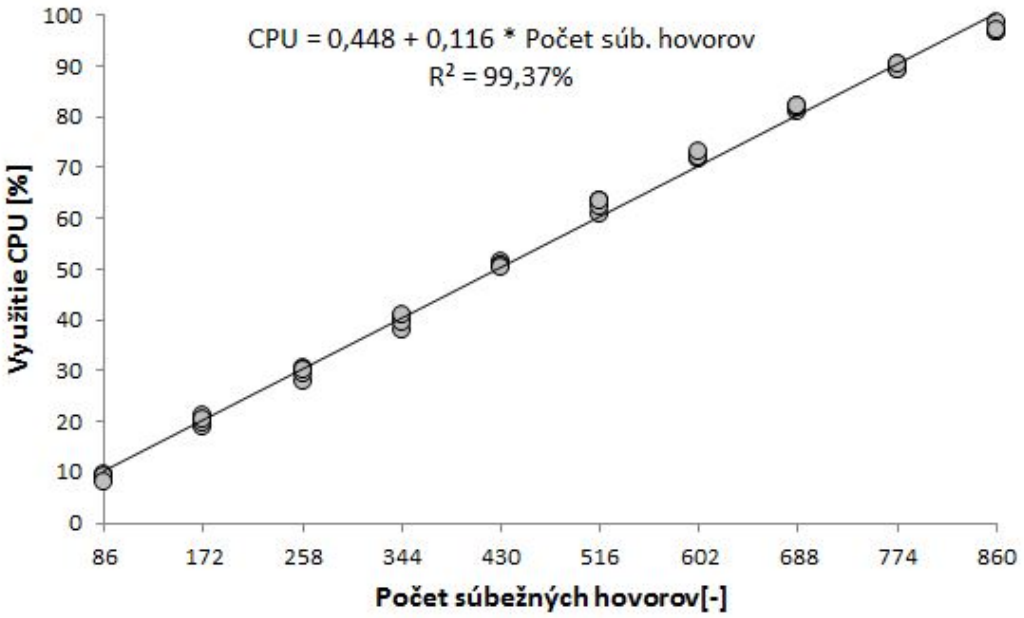
Pre každý bod som previedol 30 meraní počas 30 sekúnd. Každú sekundu jedno meranie. Na konci merania mi sysstat z týchto 30 hodnôt automatický vygeneroval priemer. Použil som príkaz pre sysstat:

```
sar -u 1 30
```

Tento postup som opakoval pre každý počet súbežných hovorov 5 krát, s tým že po každom meraní som reštartoval Kamailio aj RTPproxy. Z takto nameraných hodnôt som zostavil bodový graf a zakreslil regresnú krivku pre každý kodek. Pri testoch som sa snažil dosiahnuť hranice využitia CPU 97%, pretože tieto hodnoty sa pre každý bod pohybovali v rozmedzí +-2%. Ak by som dosiahol stavu 100% došlo by k preťaženiu CPU a ten by už nedokázal spracovať ďalšie hovory.

6.2.1 Kodek G.711

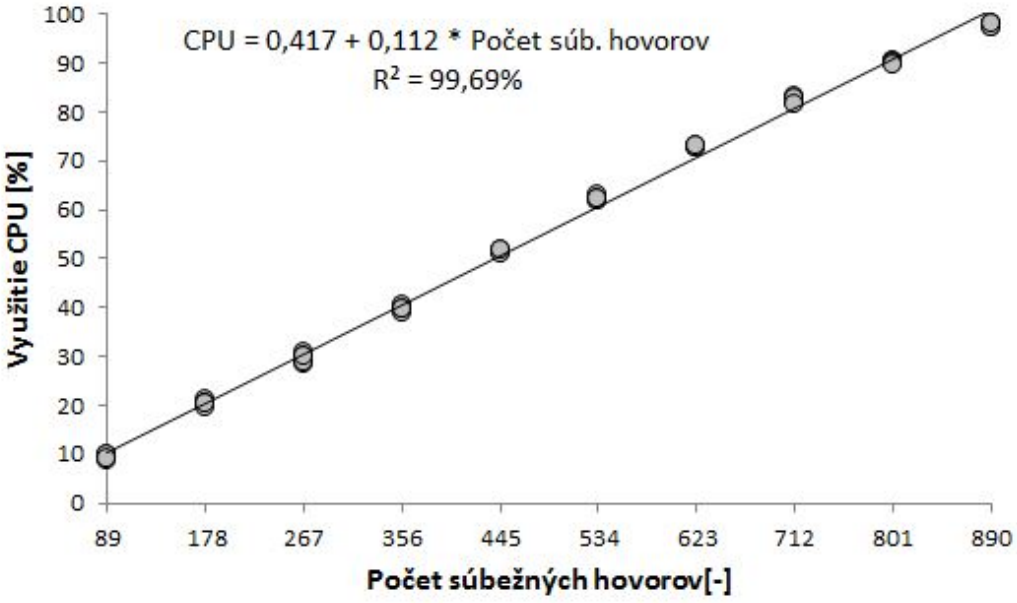
Na grafe 6.1 vidíme, vplyv RTPproxy s použitím kodeku G.711 na využitie CPU. S týmto kodekom som dosiahol maximálne 860 súbežných hovorov.



Obr. 6.1: Využitie CPU s kodekom G.711

6.2.2 Kodek G.729

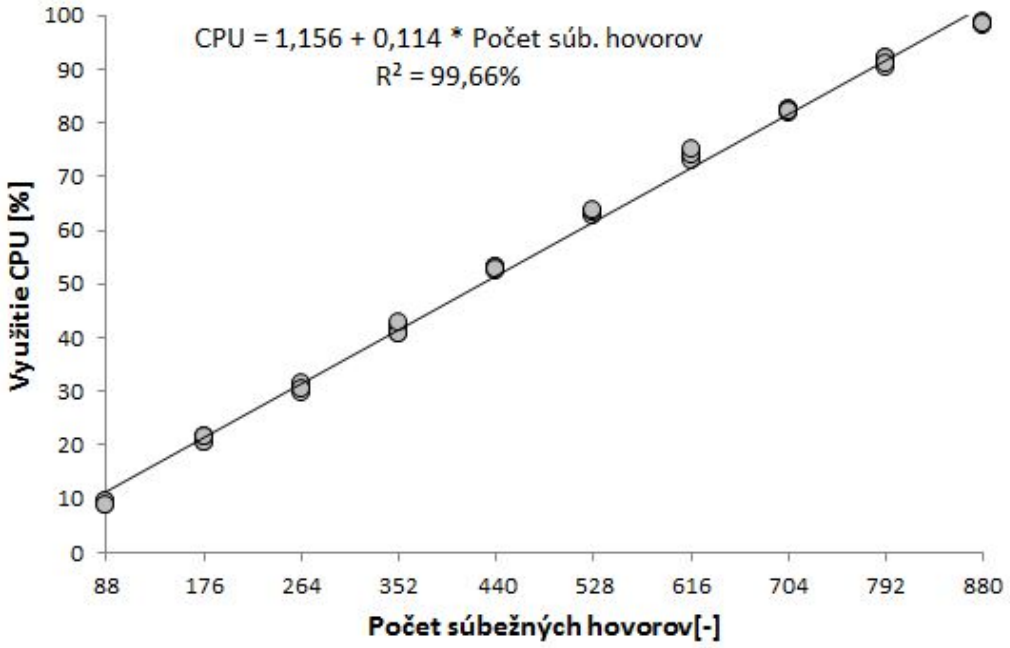
Na grafe 6.2 vidíme, vplyv RTPproxy s použitím kodeku G.729 na využitie CPU. S týmto kodekom som dosiahol maximálne 890 súbežných hovorov.



Obr. 6.2: Využitie CPU s kodekom G.729

6.2.3 Kodek G.722

Graf 6.3 zobrazuje vplyv RTPproxy s použitím kodeku G.722 na využitie CPU. S týmto kodekom som dosiahol maximálne 880 súbežných hovorov.



Obr. 6.3: Využitie CPU s kodekom G.722

7 Záver

Cieľom tejto bakalárskej práce bolo nakonfigurovať open source SIP proxy server Kamailio v spolupráci s modulom RTPproxy a zistiť, aký vplyv má toto riešenie na výkonnosť serveru. Kamailio aj RTPproxy som nainštaloval na vzdialenom virtuálnom serveri. Potom som nakonfiguroval Kamailio v spolupráci s RTPproxy. Okrem konfigurácie som musel upraviť aj smerovaciu logiku Kamailia tak, aby sa modul RTPproxy použil ako média proxy server. K tomu som použil funkciu rtpproxy_manage(), ktorá pracuje na princípe prepisovania hodnôt v SDP správach. Následne som otestoval vplyv tohto riešenia na výkonnosť serveru. Hlavne som sledoval využitie procesoru pomocou programu sysstat pri použití kodekov G.711, G.729 a G.722. Výsledky testov som spracoval vo forme grafov. Tie potvrdzujú, že použitie modulu RTPproxy má výrazný vplyv na využitie procesoru. To je predovšetkým spôsobene tým, že procesor okrem signalizačných správ musí spracovávať aj média, ktoré spôsobujú výrazné zvýšenie využitia CPU. Samostatné Kamailio tak dokáže zvládať tisíce súbežných hovorov. Rozdiel v maximálnom počte súbežných hovorov medzi jednotlivými kodekmi nebol príliš veľký. Najviac súbežných hovorov 890 som dosiahol s kodekom G.729 a najmenej 860 s kodekom G.711.

Počas konfigurácie Kamailia a RTPproxy nenastala žiadna vážna komplikácia. Problém bol až s nastavením testovacieho programu SIPp, pomocou ktorého som simuloval hovory. Implicitné scenáre SIPp nedokázali korektne pracovať s Kamailiom. Konkrétne išlo o problém s udržiavaním Kamailia v komunikácií aj počas rozpadu hovoru. Tento problém som vyriešil vytvorením vlastných scenárov. Výsledky týchto testov môžu byť užitočné najmä pri výbere vhodného serveru pre Kamailio a modul RTPproxy a tiež pri výbere kodeku. Túto prácu som zostavoval na základe čisto anglických zdrojov, preto môže byť táto práca použitá ako istý slovenský manuál pri aplikácií Kamailia v spolupráci s modulom RTPproxy.

8 Referencie

[1] Úvod do VoIP. *MB DATA* [online]. [cit. 2014-03-31]. Dostupné z: http://www.mbdata.cz/uvoddovoip.htm#_Co_je_to_VoIP

[2] VOZŇÁK, Miroslav. *Voice over IP. 1. vyd.* Ostrava: VŠB - Technická univerzita Ostrava, 2008, 176 s. ISBN 978-80-248-1828-3.

[3] Voice over Internet protocol. *Wikipedia* [online]. [cit. 2014-03-31]. Dostupné z: http://cs.wikipedia.org/wiki/Voice_over_Internet_Protocol

[4] Design and research of voice channel in VoIP system. *masters.donntu.edu.ua* [online]. [cit. 2014-05-03]. Dostupné z: http://masters.donntu.edu.ua/2012/fkita/budishevskiy/diss/indexe.htm

[5] VoIP základné pojmy a technológie. *ktl.elf.stuba* [online]. [cit. 2014-03-31]. Dostupné z: http://www.ktl.elf.stuba.sk/~halas/ntss/NTSS_VoIP_technologie.pdf

[6] Voice over IP : Protocols and Standards. *cse.wustl.edu* [online]. [cit. 2014-03-31]. Dostupné z: http://www.cse.wustl.edu/~jain/cis788-99/ftp/voip_protocols.pdf

[7] Alan B., Johnston. *SIP: Understanding the Session Initiation Protocol Third Edition.* Boston,Londýn, ARTECH HOUSE, 2009, 427 s., ISBN 13: 978-1-60783-995-8.

[8] Flavio E., Goncalves. *Building Telephony Systems with OpenSER.* Birmingham, Packt Publishing Ltd., 2008, 324 s., ISBN 13: 978-1-847193-73-5.

[9] Studium protokolu Session Decription Protocol. *cs.vsb* [online]. [cit. 2014-04-02]. Dostupné z: http://www.cs.vsb.cz/grygarek/TPS/projekty/0607Z/SDP.pdf

[10] Streaming media (4): transportní protokoly RTP/RTCP. *dsl.cz* [online]. [cit. 2014-04-02]. Dostupné z: http://www.dsl.cz/clanek/60-streaming-media-4-transportni-protokoly-rtp-rtcp

[11] DEB Packages. *kamailio.org* [online]. [cit. 2014-05-01]. Dostupné z: http://www.kamailio.org/wiki/packages/debs

[12] Kamailio SIP Server v3.2.0 Development Guide. *asipto.com* [online]. [cit. 2014-04-03]. Dostupné z: http://www.asipto.com/pub/kamailio-devel-guide/#id36131094

[13] Welcome to Kamailio™ – the Open Source SIP Server. *kamailio.com* [online]. [cit. 2014-04-03]. Dostupné z: http://www.kamailio.org/w/

[14] About Sippy RTPproxy. *rtpproxy.org* [online]. [cit. 2014-04-05]. Dostupné z: http://www.rtpproxy.org/

[15] RTPProxy. *voip-info.org* [online]. [cit. 2014-04-05]. Dostupné z: http://www.voip-info.org/wiki/view/RTPProxy

[16] rtpproxy Module. *kamailio.org* [online]. [cit. 2014-04-06]. Dostupné z: http://kamailio.org/docs/modules/4.1.x/modules/rtpproxy.html

[17] SER module mediaproxy. *voip-info.org* [online]. [cit. 2014-04-06]. Dostupné z: http://www.voip-info.org/wiki/view/SER+module+mediaproxy

[18] Mediaproxy Module. *kamailio.org* [online]. [cit. 2014-04-06]. Dostupné z: http://kamailio.org/docs/modules/stable/modules/mediaproxy.html

[19] SIPp reference documentation. *sipp.sourceforge.net* [online]. [cit. 2014-04-17]. Dostupné z: http://sipp.sourceforge.net/doc/reference.html

A Obsah priloženého CD

1. Text bakalárskej práce vo formáte PDF, súbor: bakalarska_ praca.pdf
2. Konfiguračný súbor Kamailia, súbor: kamailio.cfg
3. SIPp scenár UAC, súbor: UAC.xml
4. SIPp scenár UAS, súbor: UAS.xml
5. Pcap súbor pre kodek G.711, súbor: g711A.pcap
6. Pcap súbor pre kodek G.729, súbor: g729.pcap
7. Pcap súbor pre kodek G.722, súbor: g722.pcap